International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 11 | Nov - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

Internet of Things-Based Smart Greenhouse with Artificial Intelligence-Assisted Plant Health Monitoring Using ESP32-CAM

Akshatha B L, Nithya M V, Niveditha B M, Rakshitha S G
Department of Electronics and Communication Engineering, Kalpataru Institute of Technology, Tiptur, Karnataka, India

---

Abstract - Manual monitoring of soil moisture, climate and plant health in greenhouses is time-consuming, and most low-cost systems do not combine environmental automation with plant health analysis. This paper presents a low-cost Internet of Things (IoT) smart greenhouse in which an ESP32-CAM acts as the main controller, running the control logic, hosting a Wi-Fi web dashboard, streaming live video and capturing plant images. An Arduino Nano serves as a general-purpose input/output (GPIO) expander that interfaces temperature/humidity, soil-moisture and flame sensors, a 16×2 liquid crystal display (LCD), a buzzer, and a relay module driving a water pump and lighting, and exchanges data with the ESP32-CAM over a serial link. Captured images are sent to a Python/Flask server for artificial intelligence (AI)-based analysis of visible leaf symptoms. Test procedures were defined for every subsystem. On the prototype, Wi-Fi dashboard access, live camera streaming and AI-based analysis of a captured image were demonstrated, while sensor-data communication and soil-moisture interpretation were found to need correction and re-verification. The modular design provides an affordable foundation for greenhouse automation, with the AI output serving as decision support rather than a definitive diagnosis.

Key Words: Internet of Things, smart greenhouse, ESP32-CAM, automatic irrigation, plant health monitoring, GPIO expander.

1. INTRODUCTION

Agriculture supplies food, raw materials, employment and economic stability, and rising demand is pushing farmers toward technologies that raise productivity while conserving resources and labour. Traditional farming relies on manual observation of soil moisture, temperature, humidity and plant health, which is slow and inconsistent. Greenhouse cultivation addresses part of this problem by providing a controlled environment in which temperature, humidity, light and soil moisture can be regulated, and recent work shows that connecting such environments to Internet of Things (IoT) sensing and control improves both monitoring and resource use [1], [2].

Environmental automation alone does not reveal the condition of the plants themselves. Diseases and nutrient stress often appear first as visible changes in leaves, and image-based deep learning has become a promising way to detect them automatically [3]. Most low-cost greenhouse prototypes, however, address either environmental monitoring or image analysis, but not both within one affordable system.

This paper presents an IoT smart greenhouse that integrates the two. An ESP32-CAM acts as the main controller, providing control logic, Wi-Fi connectivity, live video and image capture, while an Arduino Nano serves as a general-purpose input/output (GPIO) expander for the sensors and actuators and a Python-based server performs artificial intelligence (AI)-assisted plant health analysis. The main contributions are:

- a low-cost, modular architecture that combines environmental sensing, relay-based actuation, local display, fire alerting and camera-based plant health analysis in one prototype;
- a soil-moisture-driven irrigation scheme with a bounded pump run time and a re-trigger lock-out to avoid over-watering;
- a Wi-Fi web dashboard that unifies sensor data, live video and AI analysis results; and
- a documented hardware interface, including pin mapping and 5 V/3.3 V serial level handling, that others can reproduce.

The rest of the paper is organised as follows. Section 2 reviews the literature and identifies research gaps. Section 3 states the problem. Section 4 describes the proposed methodology, including the system design, hardware and software. Section 5 presents testing, results and discussion, and Section 6 concludes with future scope.

2. LITERATURE REVIEW

2.1 Internet of Things-Based Greenhouse Monitoring and Control

Thomopoulos *et al.* combine IoT sensing with fuzzy-logic control in a small-scale smart greenhouse, showing the value of intelligent decision-making over fixed rules, although fuzzy control demands careful rule design and accurate sensor readings [4]. Folorunso *et al.* develop an IoT-based greenhouse that integrates sensors and electronic control to reduce manual work and maintain suitable conditions [5]. Tomar and Saroha describe real-time monitoring and automated control for efficient greenhouse agriculture [6]. Méndez-Guzmán *et al.* apply IoT monitoring of climatic conditions and crop appearance to an aeroponics greenhouse [7], and Hernández-Morales *et al.* report a practical monitoring system for protected cultivation, stressing sensor placement and reliable communication [8]. More recent studies extend this line to IoT-based fertigation scheduling with wireless microclimate monitoring [1] and IoT-enhanced decision support for real-time microclimate monitoring and control [2].

2.2 Artificial Intelligence-Based Plant Disease Detection

Saleem *et al.* compared three deep-learning meta-architectures, Single Shot MultiBox Detector (SSD), Faster Region-based Convolutional Neural Network (Faster R-CNN) and Region-based Fully Convolutional Network (R-FCN), for detecting and classifying healthy and diseased leaves in images; SSD combined with the Adam optimiser achieved the highest mean average precision (mAP) of 73.07% [9]. Related work includes a survey of computer vision for plant disease monitoring in precision agriculture [10], deep-learning disease detection models for plants [11], reviews of deep-learning plant disease detection [12], [15], a hybrid vision transformer and convolutional neural network (CNN) classifier [13], a deep residual CNN for leaf disease detection [14], and a CNN with mobile-app integration for early detection [3]. These approaches require suitable image datasets, model training and computing resources, and they do not control the greenhouse environment.

2.3 Research Gaps and Positioning

The literature reveals five gaps that motivate this work. First, environmental monitoring and plant health analysis are seldom integrated in a single low-cost system. Second, plant inspection

is still mostly manual, and automated image analysis depends on model choice and image quality. Third, fixed irrigation schedules ignore actual soil moisture and can waste water. Fourth, many low-cost systems offer only a local display or only remote monitoring, not both. Fifth, advanced prototypes tend to be built for a single function, so adding sensors or actuators requires substantial redesign. The proposed system addresses these gaps with soil-moisture-based irrigation, a local LCD plus a web dashboard, and a modular split between environmental control and image processing.

3. PROBLEM STATEMENT

Conventional greenhouse management depends on manual monitoring: operators periodically inspect plants and measure conditions, and irrigation follows fixed schedules or visual judgement. This demands continuous human supervision, may cause excessive or insufficient watering, can miss environmental changes, lets plant diseases go unnoticed in early stages, and rarely offers remote monitoring. The problem addressed here is to design and develop an IoT-based smart greenhouse that continuously monitors environmental parameters, automatically controls irrigation and lighting, provides local monitoring and safety indications, and incorporates AI-based plant image analysis to support plant health monitoring.

4. PROPOSED METHODOLOGY

4.1 System Overview

The system comprises eight cooperating units: sensor monitoring, Arduino Nano GPIO expansion, ESP32-CAM main control, communication and imaging, automatic irrigation control, liquid crystal display (LCD) unit, fire detection and alert, a web-based dashboard, and AI-based plant health analysis. The ESP32-CAM is the main controller: it runs the control logic, provides wireless connectivity, allows the greenhouse to be monitored from a web browser, and captures plant images that a Python-based server passes to an AI analysis service. Because the ESP32-CAM exposes few free GPIO pins, an Arduino Nano is used purely as a GPIO expander that reads the sensors and drives the LCD, buzzer and relays on behalf of the ESP32-CAM.

4.2 Hardware Architecture

Figure 1 shows the block diagram. The architecture has three sections. The input section comprises a DHT11 temperature/humidity sensor, a soil-moisture sensor with an LM393 comparator module, and a flame sensor. The I/O expansion section is the Arduino Nano, which provides additional GPIO interfaces for the connected sensors, LCD, buzzer and 4-channel relay module. It reads the sensor signals and exchanges the corresponding data with the ESP32-CAM. The ESP32-CAM is the main control and monitoring unit, which executes the control logic and sends the required actuator commands to the Nano. The main control and monitoring section is the ESP32-CAM, which communicates with the Nano over a universal asynchronous receiver–transmitter (UART) link, executes the control logic, serves the web dashboard over Wi-Fi, and sends captured images to the Python AI server running on a laptop or personal computer. A single 5 V DC adapter powers all modules from a common supply rail. Lighting control is included through the relay interface, while any LDR-based light sensing should be treated as an optional extension unless implemented and verified in the final prototype.

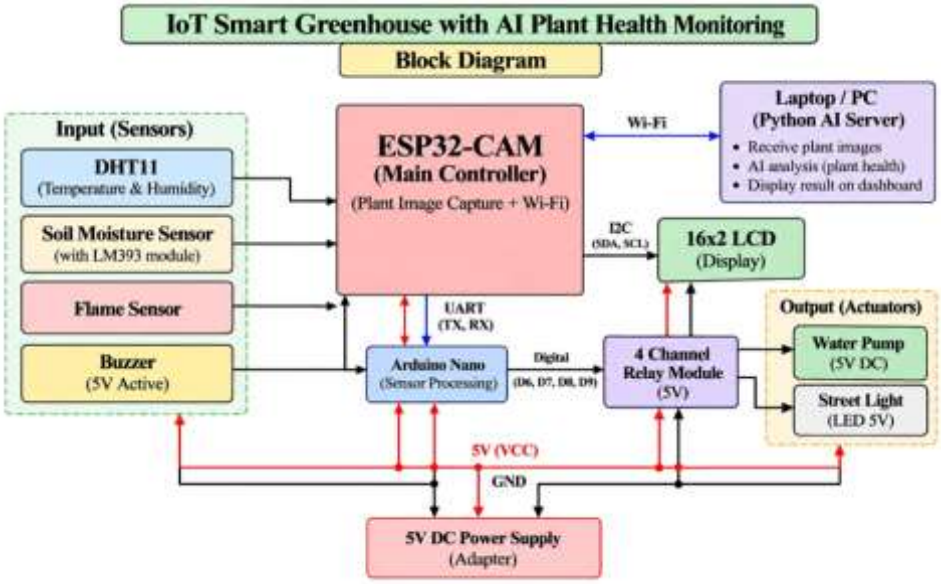


Fig -1: Block diagram of the IoT smart greenhouse with AI plant health monitoring.

4.3 Interfacing and Circuit Connections

The DHT11 data line, the digital outputs of the soil-moisture and flame sensors, the lighting control line, the pump relay input and the buzzer are connected to Arduino Nano digital pins as listed in Table 1. The LCD uses an inter-integrated circuit (I²C) interface through the SDA and SCL lines on A4 and A5. Serial communication with the ESP32-CAM uses two digital pins configured as a software serial port (Table 2); the full circuit is shown in Fig. 2.

Table -1: Arduino Nano pin assignment

Component	Nano pin
DHT11 data	D2
Soil-moisture digital output	D3
Flame-sensor digital output	D4
Lighting (LED) control	D5
Water-pump relay control	D6
Buzzer	D7
ESP32-CAM link, RX	D10
ESP32-CAM link, TX	D11
I²C LCD SDA / SCL	A4 / A5

Table -2: Nano to ESP32-CAM serial connections

Arduino Nano	ESP32-CAM	Remark
D10 (RX)	GPIO13 (TX)	Direct
D11 (TX)	GPIO14 (RX)	Level reduction
GND	GND	Common ground

Two electrical points are critical. The controllers must share a common ground, and because the 5 V Nano transmit line drives the 3.3 V ESP32-CAM receive input, voltage-level reduction is required on that line to protect the ESP32-CAM. Relay contacts (COM and NO) isolate the pump circuit from the control electronics.

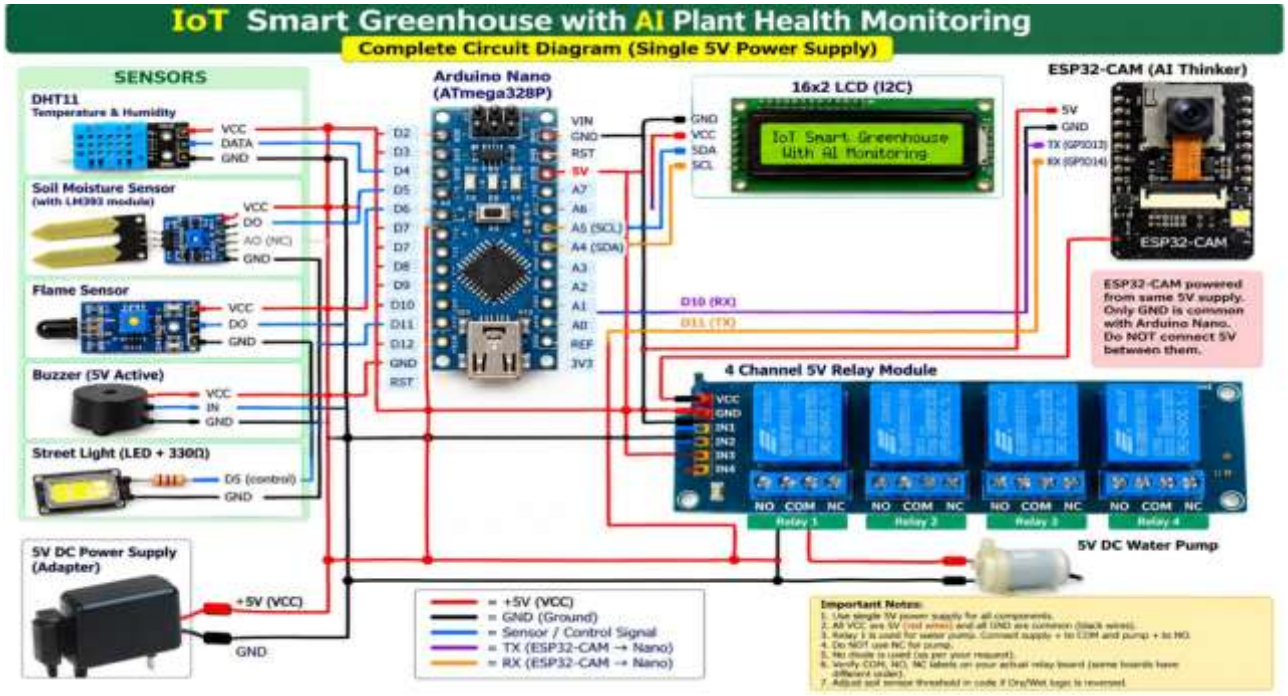


Fig -2: Complete circuit diagram of the system with a single 5 V power supply.

4.4 Working Principle

On power-up, the ESP32-CAM initializes the main system and joins the configured Wi-Fi network, while the Arduino Nano initializes the GPIO interfaces connected to the sensors and LCD. In each cycle, the Nano reads the connected sensor signals, updates the LCD, and exchanges the sensor data with the ESP32-CAM over serial. The ESP32-CAM compares the readings with its thresholds, executes the primary control logic, updates the dashboard, and sends back pump, buzzer and lighting commands, which the Nano applies to its output pins. The irrigation and alert logic therefore runs locally on the ESP32-CAM, independent of internet connectivity and the AI service. In parallel, the ESP32-CAM streams live video and captures plant images; when analysis is requested, an image is submitted to the Python AI server and the result is shown on the dashboard. The operation is summarised by the flowchart in Fig. 3, and the process repeats until monitoring is stopped.

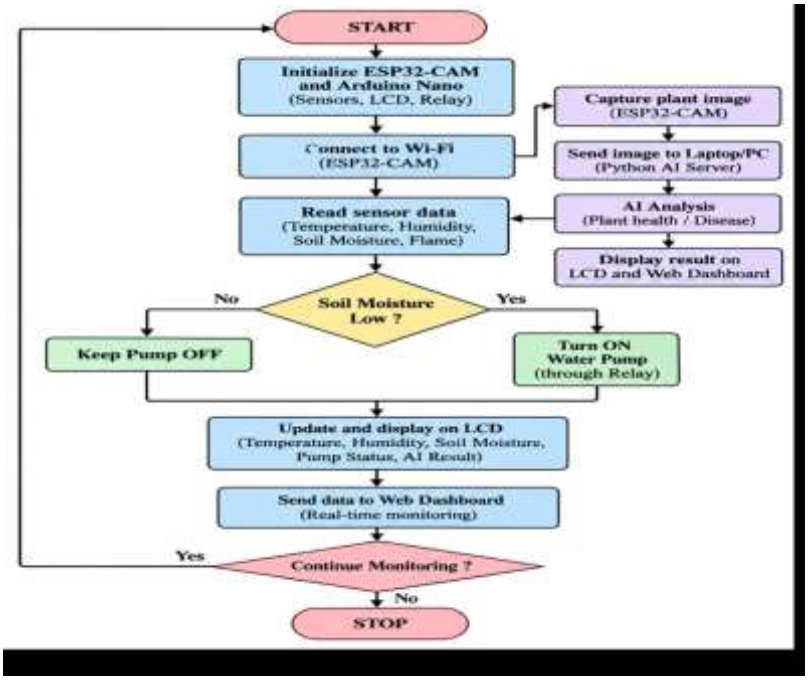


Fig -3: Flowchart of smart greenhouse operation.

4.5 Automatic Irrigation Control

Irrigation is the principal automatic function. The digital output of the soil-moisture sensor, read by the Nano and reported to the ESP32-CAM, is interpreted as WET or DRY according to calibrated sensor logic. The algorithm is:

1) Read the soil-moisture sensor and classify the soil as WET or DRY.
2) If WET, keep the pump OFF.
3) If DRY, check whether a new automatic irrigation cycle is permitted.
4) If permitted, command the Nano to energise the relay and start the pump.
5) Stop the pump when the soil becomes WET or the maximum run time is reached.
6) Wait for the conditions that re-enable another automatic cycle.

The pump runs for at most five seconds per permitted cycle, and repeated activation is blocked while the same dry condition persists, which prevents over-watering and unnecessary water use. Reliable operation depends on proper calibration of the sensor threshold, and manual pump control is available through the dashboard where enabled.

4.6 Fire Detection and Alert

The flame-sensor output is read by the Nano and reported to the ESP32-CAM; when a possible flame is detected, the buzzer is activated through the Nano's output according to the programmed alert logic to warn nearby users. The flame sensor is a supplementary warning device and not a substitute for a certified fire-safety system.

4.7 ESP32-CAM, Dashboard and Streaming

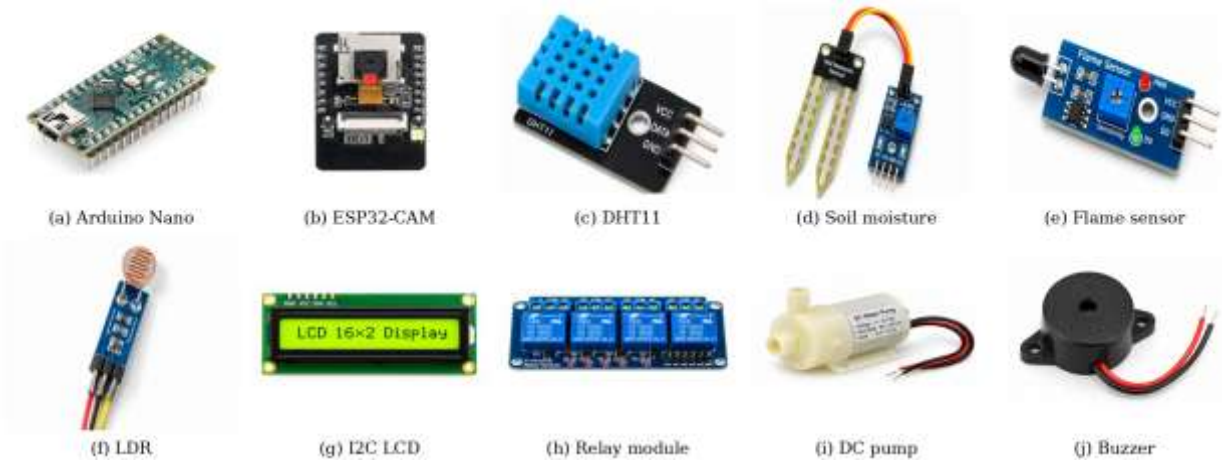
The ESP32-CAM hosts a web interface with sections for environmental monitoring, irrigation status, live camera view and plant health analysis. Sensor values received over serial are displayed alongside the stream, so users can inspect plant condition remotely without visiting the greenhouse. Still images captured by the camera are the input to the AI module, and image capture operates independently of the routine sensor-monitoring loop.

4.8 Artificial Intelligence-Based Plant Health Analysis

Images from the ESP32-CAM, or photos uploaded by the user, are sent to a Python/Flask server, which forwards them through the configured workflow to an AI image-analysis service. The service examines visible symptoms such as leaf discolouration, damaged areas and other abnormalities and returns a structured text report, which the server displays on the dashboard. The dashboard supports capture-and-analyse from the live camera and upload of one to five photographs for individual or combined analysis. The result is assistive information only and must not be treated as a guaranteed disease diagnosis.

4.9 Hardware Components

The components used are shown in Fig. 4. The ESP32-CAM is the main controller; it integrates Wi-Fi and a camera, runs the control logic, serves the dashboard, streams video and interfaces the greenhouse hardware with the external AI system. Because it has few free GPIO pins, a compact Arduino Nano (ATmega328P) is used as a GPIO expander, providing the digital, analogue and I²C pins needed for the sensors, LCD, buzzer and relays. The DHT11 digital sensor measures ambient temperature and relative humidity, which strongly influence plant growth. The soil-moisture sensor supplies a digital WET/DRY output. The flame sensor detects radiation from a flame and signals the Nano. A light-dependent resistor (LDR), whose resistance varies with incident light, supports day/night detection for lighting control. A 16×2 character LCD with an I²C backpack minimises the pins needed for local display. A 4-channel 5 V relay module lets the low-power controller switch the DC water pump and lighting loads, and an active buzzer provides audible warning.



(a) Arduino Nano (b) ESP32-CAM (c) DHT11 (d) Soil moisture (e) Flame sensor
(f) LDR (g) I2C LCD (h) Relay module (i) DC pump (j) Buzzer

Fig -4: Hardware components: (a) Arduino Nano, (b) ESP32-CAM, (c) DHT11, (d) soil-moisture sensor, (e) flame sensor, (f) LDR, (g) I²C LCD, (h) relay module, (i) DC water pump, (j) buzzer.

4.10 Software Tools

Firmware for the Arduino Nano and ESP32-CAM is written in embedded C/C++ using the Arduino Integrated Development Environment (IDE), which also provides the serial monitor for testing and debugging. The Nano firmware handles GPIO interfacing, sensor acquisition, LCD operation, relay and buzzer actuation, and serial communication. The ESP32-CAM firmware implements the primary control logic, Wi-Fi communication, web server functionality, dashboard interaction, and camera operation. The AI server is written in Python using the lightweight Flask framework, which provides the Hypertext Transfer Protocol (HTTP) endpoints for image upload and returns analysis results to the interface. The dashboard is built with HyperText Markup Language (HTML) for structure, Cascading Style Sheets (CSS) for styling and

JavaScript for updating readings, handling user actions and communicating with the ESP32-CAM.

4.11 Functional and Non-Functional Requirements

Functionally, the system periodically reads temperature and humidity, detects soil-moisture state, shows readings on the LCD and dashboard, forwards data to the ESP32-CAM, operates the pump automatically (with optional manual control), detects flame and sounds an alert, provides live camera monitoring, and captures plant images for AI analysis. Non-functionally, it must be reliable, usable, safe, maintainable and affordable: readings should be consistent and respond to changes within a reasonable time, the dashboard should open in a standard browser on the connected network, modules should be testable independently, and wiring must respect the operating voltages of each device.

5. RESULTS AND DISCUSSION

5.1 Prototype

The prototype (Fig. 5) places the control unit—ESP32-CAM on a breadboard, Arduino Nano I/O expander, relay module and LCD—beside a plant zone with potted plants, the soil-moisture and DHT11 sensors, a reservoir and pump with a delivery tube, and a lamp, all under a transparent greenhouse cover. It is designed to provide real-time monitoring of temperature, humidity and soil moisture on the LCD and dashboard, automatic pump control through the relay, buzzer alerts on flame detection, live camera monitoring, and AI analysis of captured images.



Fig -5: Prototype of the smart greenhouse monitoring system.

5.2 Subsystem and Integrated Testing

A test procedure was defined for each subsystem (Table 3), followed by integrated testing in which sensor changes are observed simultaneously on the LCD and dashboard while checking for communication interruptions and misinterpreted readings. During development, firmware upload to the Nano, Wi-Fi connection and camera streaming were demonstrated. A later firmware change, however, caused the dashboard to show sensor readings as unavailable, and the soil-moisture sensor output was at one point interpreted incorrectly. Both issues must be corrected and the tests repeated before measured results can be reported.

Table -3: Test procedures and verification criteria

Subsystem	Procedure	Criterion
DHT11	Connect to Nano; read values on serial monitor and LCD	Correct initialisation and in-range data
Soil sensor	Record output in dry and moist soil	Correct digital polarity in firmware
Pump	Simulate wet/dry soil; monitor relay	Runs only when permitted; stops on stop condition
Flame sensor	Safe simulated flame; observe buzzer	Buzzer sounds on detection
LCD	Vary sensor conditions	Readable, correctly updated text
Dashboard	Open ESP32-CAM IP in a browser; compare with Nano	Readings match; controls accessible
Camera	Open live stream in browser	Continuous, uninterrupted video
AI analysis	Submit captured images to server	Response relevant to visible condition

Flame testing is to be performed using safe procedures, without exposing the electronics to an uncontrolled flame. Network stability and power supply were also checked during camera testing, because both strongly affect stream quality.

5.3 Artificial Intelligence Analysis Output

Figure 6 shows the dashboard after an image was submitted. The system returned a structured report giving the health status, plant identification, possible issue, observations, advice and image count. For the test image, a single green leaf, the report stated that the plant appeared healthy with no obvious visible issue, and noted that the low image resolution limited its ability to resolve fine details. This illustrates that analysis quality depends on image quality and lighting, and it confirms the end-to-end path from ESP32-CAM or uploaded photo to the Flask server, the AI service and the dashboard.



Fig -6: Dashboard showing the AI plant health analysis result.

5.4 Discussion

The development demonstrated the feasibility of combining an ESP32-CAM, an Arduino Nano I/O expander and a Python-based AI server for greenhouse monitoring. Separating I/O interfacing on the Nano from control, networking and imaging on the ESP32-CAM and the Python server means that irrigation and fire alerts, which run on the ESP32-CAM, continue to work when internet access or the AI service is unavailable. Development also highlighted three challenges: sensor calibration, reliable serial communication between the 5 V and 3.3 V devices, and stable network operation, which together underline the importance of consistent communication protocols and integrated testing. The system should be re-

evaluated after the communication and soil-moisture issues are corrected, with quantitative measurement of irrigation behaviour, sensor error, dashboard latency and AI reliability.

5.5 Applications and Benefits

The system suits small greenhouse installations, home gardening, plant nurseries, educational laboratories demonstrating embedded systems, IoT communication and AI, and agricultural research prototypes. Its architecture can be extended into more advanced monitoring systems by adding sensors and control algorithms. Its benefits are reduced manual observation, irrigation based on measured soil condition, combined environmental and visual monitoring, both local and wireless access, and a foundation for AI-assisted plant health assessment. By watering according to soil state and shortening routine inspection, it can support better water management and earlier identification of visible plant problems.

5.6 Limitations

Because the Nano only relays I/O, control also depends on the serial link between the two boards. The DHT11 offers only basic accuracy and is unsuitable for precision applications. The digital soil-moisture sensor needs calibration for reliable WET/DRY classification and gives no continuous moisture value. Remote dashboard access depends on stable Wi-Fi, and AI analysis requires a running Python server and access to the configured AI service. Analysis can be degraded by lighting, camera quality and image clarity. The flame sensor is not a certified fire-safety device. Resistive soil probes corrode over time, mechanical relays have a finite switching life (so threshold hysteresis and minimum switching intervals are advisable), and the pump draws far more current than the controllers, so an adequately rated power supply is essential. As a prototype, the system needs further testing and validation before use in critical or large-scale agricultural environments.

6. CONCLUSION AND FUTURE SCOPE

This paper presented an IoT smart greenhouse that unifies environmental monitoring, soil-moisture-based irrigation, fire alerting, a local LCD, a Wi-Fi web dashboard with live video, and AI-assisted plant health analysis, using an ESP32-CAM as the main controller, an Arduino Nano as a GPIO expander, and a Python/Flask server. The modular, low-cost design provides a practical foundation for more intelligent greenhouse management.

Future scope includes more accurate sensors, larger plant-image datasets to improve disease identification and early detection, cloud storage of sensor and plant health records, and a mobile application with alerts for abnormal temperature, fire and soil moisture. A calibrated capacitive soil-moisture sensor with analogue output would allow more flexible irrigation control than digital WET/DRY detection, and automatic temperature control with a cooling fan, data logging, solar power and multi-section greenhouse expansion are further extensions. A systematic quantitative evaluation of the integrated prototype under different operating conditions is also planned.

ACKNOWLEDGEMENT

The authors thank their project guide, Sunil Kumar K M, and the Department of Electronics and Communication Engineering, Kalpataru Institute of Technology, Tiptur, for guidance and laboratory support.

REFERENCES

[1] M. Hasan *et al.*, “IoT-based smart fertigation scheduling and wireless microclimate monitoring for a greenhouse Dutch bucket hydroponic system,” *Irrigation and Drainage*, 2025.

[2] D.-I. Săcăleanu, M.-G. Matache, Ș.-G. Roșu, B.-C. Florea, I.-P. Manciu, and L.-A. Perișoară, “IoT-enhanced decision support system for real-time greenhouse microclimate monitoring and control,” *Technologies*, 2024.

[3] M. Iftikhar, I. A. Kandhro, N. Kausar, A. Kehar, M. Uddin, and A. Dandoush, “Plant disease management: A fine-tuned enhanced CNN approach with mobile app integration for early detection and classification,” *Artificial Intelligence Review*, 2024.

[4] V. Thomopoulos, F. Tolis, T.-F. Blounas, D. Tsipianitis, and A. Kavga, “Application of fuzzy logic and IoT in a small-scale smart greenhouse system,” *Smart Agricultural Technology*, 2024.

[5] T. A. Folorunso, O. Oshiga, J. A. Bala, and D. A. Ojogunwa, “Development of an Internet of Things based smart greenhouse,” *International Journal of Computer Trends and Technology*, 2024.

[6] V. Tomar and N. Saroha, “IoT-enabled smart greenhouse: Real-time monitoring and automated control for efficient agriculture,” *International Journal of Research and Review in Applied Science, Humanities, and Technology*, 2024.

[7] H. A. Méndez-Guzmán *et al.*, “IoT-based monitoring system applied to aeroponics greenhouse,” *Sensors*, 2022.

[8] C. A. Hernández-Morales, J. M. Luna-Rivera, and R. Perez-Jimenez, “Design and deployment of a practical IoT-based monitoring system for protected cultivations,” *Computer Communications*, 2022.

[9] M. H. Saleem, S. Khanchi, J. Potgieter, and K. M. Arif, “Image-based plant disease identification by deep learning meta-architectures,” *Plants*, vol. 9, no. 11, Art. no. 1451, 2020.

[10] W. Ding, M. Abdel-Basset, I. Alrashdi, and H. Hawash, “Next generation of computer vision for plant disease monitoring in precision agriculture: A contemporary survey, taxonomy, experiments, and future direction,” *Information Sciences*, 2024.

[11] M. Jung *et al.*, “Construction of deep learning-based disease detection model in plants,” *Scientific Reports*, 2023.

[12] M. Shoaib *et al.*, “An advanced deep learning models-based plant disease detection: A review of recent research,” *Frontiers in Plant Science*, 2023.

[13] P. S. Thakur, S. Chaturvedi, P. Khanna, T. Sheorey, and A. Ojha, “Vision transformer meets convolutional neural network for plant disease classification,” *Ecological Informatics*, 2023.

[14] J. Arun Pandian, K. Kanchanadevi, N. R. Rajalakshmi, and G. Arulkumaran, “An improved deep residual convolutional neural network for plant leaf disease detection,” *Computational Intelligence and Neuroscience*, 2022.

[15] J. Liu and X. Wang, “Plant diseases and pests detection based on deep learning: A review,” *Plant Methods*, 2021.